INTEGRACIÓN DE UNA CÁMARA PAN-TILT SUBACUÁTICA PARA DETECCIÓN DE OBJETOS Y GUIADO DE UN ROV SUBMARINO

C. Veiga Almagro, J.C. García Sánchez, D. Fornas García, R. Marín Prades, Pedro J. Sanz
Universitat Jaume I
{carlos.veiga, garciaju, dfornas, rmarin, sanzp}@uji.es

Resumen

En este documento se expone un sistema adaptativo y semi-autónomo, el cual genera una interfaz de alto nivel para facilitar el trabajo de supervisión humana, capaz de detectar un objeto de interés, con la finalidad de trackear su posición y guiar a un ***Remote Operated Vehicle (ROV)*** *haciendo uso de una cámara pan-tilt submarina, mediante la utilización de herramientas conocidas de visión por computador, y a través del middleware Robot Operating System (ROS). El trabajo se ha llevado a cabo en el contexto del proyecto coordinado MERBOTS (www.irs.uji.es/merbots), para mejorar la realimentación visual en intervenciones reales submarinas.*

Palabras clave: Robótica submarina, intervenciones robóticas, visión por computador, detección de objetos

1. Introdución

Una de las tareas más importantes dentro del ámbito de la robótica de intervención, es la identificación/detección de los elementos de interés con los que la plataforma debe de interactuar. Esto se convierte en una tarea crítica (junto con las de aproximación y agarre) dentro de la intervención que sea necesario llevar a cabo. Estas tareas, incrementan su complejidad cuando el contexto es llevado debajo del agua, donde inconvenientes tales como la deformación, la reflexión, la aparición de ruido (gaussiano, granular, etc.), la ocultación parcial (o incluso completa) de los objetos de interés, y otros agentes como puede ser la suspensión de partículas en el agua (e.g. tierra, algas, etc.), o el movimiento de las corrientes marinas, dificultando el desarrollo de comportamientos 100 % autónomos. Todo ese conjunto de premisas enumeradas en las líneas anteriores, genera la necesidad de desarrollar sistemas semi-autónomos, donde exista una supervisión humana capaz de apoyar en la búsqueda y detección de los elementos de interés, y hacer así frente a la tesitura.

Figura 1: Cámara pan-tilt EAC-DTR100ZC

Una de las principales necesidades a la hora de dirigir un Vehículo Operado Remotamente (ROV), es tener un control total del entorno en el que se pretende realizar la operación. Es bien conocido que en el ámbito marino, la localización mediante dispositivos tipo sonar (pasivo o activo), es una de las principales herramientas para detección y evasión de colisión de los diferentes obstáculos que se puedan encontrar sumergidos bajo la superficie del agua, que si bien, proporcionan una reconstrucción tridimensional del mapa marino[3], no están exentos de un alto índice de ruido, a la vez que puede afectar al comportamiento de la fauna que habita en el entorno del radio de acción, como se muestra en[5], ya sea emitiendo tanto ondas electromagnéticas como acústicas. A parte de estas ventajas e inconvenientes, se encuentra la necesidad que nos atañe en este documento, la detección de objetos de interés (2D o 3D), tanto de forma autónoma como supervisada [10], donde las cámaras son sensores especialmente adecuados.

2. ADAPTACIÓN DE UNA CÁMARA PAN-TILT SUBMARINA

2.1. Requisitos del Sistema

El uso de cámaras resulta necesario en cuanto a la tarea de texturizar los obstáculos que comprenden el mapeo del fondo marino. Para llevar a cabo este tipo de tareas, los sistemas dotados de cámaras (pares estéreo o sistemas con láser), se convierten en una herramienta fundamental, capaz de recrear el entorno de forma tridimensional, a la vez que dota de texturas al mismo [9].

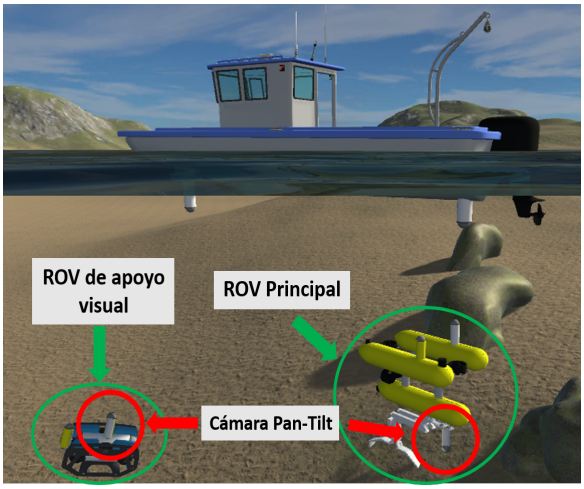

Figura 2: Representación escena con nuestra cámara en ROVs de distinta utilidad

El reto que se afronta en este árticulo va un paso más allá, donde haciendo uso de una cámara pan-tilt submarina, la cual ha sido integrada en el entorno del middleware ROS [4], ampliamente utilizado en las distintas plataformas robóticas desarrolladas, y que es la base del desarrollo de gran parte de los sistemas creados en el Interactive and Robotic Systems Lab (IRSLab). La cámara ofrece una gran libertad de movimiento tanto en horizontal como en vertical, permitiendo una inspección visual completa del entorno. Se pretende crear un sistema capaz de detectar objetos, aplicar un seguimiento a los mismos, y guiar el ROV a alcanzar los objetivos especificados por el usuario (ver figura 2).

Así mismo, es importante destacar el desafío que representa el uso de una cámara en un entorno submarino, añadiendo complejidad respecto a su uso en condiciones normales, como puede ser: la falta de estabilidad del vehículo portador, o la reducción de visibilidad debido a la dispersión de partículas en el agua. Esto acarrea la necesidad de hacer uso de herramientas de visión por computador con la intención de suavizar los inconvenientes anteriormente descritos.

2.1.1. Recursos Utilizados

La cámara Pan-Tilt utilizada es el modelo ECA-DTR100ZC, tal cual se puede observar en la figura 1: con un ángulo de giro horizontal de 360º y otro +-90º de giro vertical. Tiene un peso de 4.8kg en el aire que se ve reducido a 2.3kg cuando está sumergida en el agua. Dispone de un zoom óptico de 10 aumentos, soportando hasta 30 atmósferas de presión (300 metros de profundidad) y un rango de temperaturas que va desde -20ºC hasta 60ºC.

3. Descripción del Sistema

El sistema incluye una interfaz de bajo nivel que permite la teleoperación de la cámara pan-tilt, por medio de la arquitectura software que se puede observar en la figura 3. Esta arquitectura integra la cámara subacuática en el entorno ROS permitiendo al usuario capturar imágenes e interaccionar con ella tanto a nivel teleoperado como haciendo uso de comportamientos autónomos.

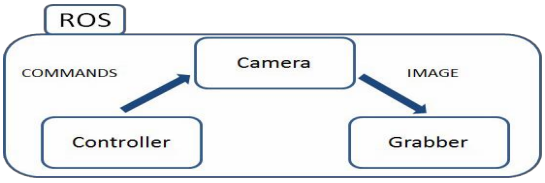

Figura 3: Estado inicial de la Arquitectura software del sistema

3.1. Ecualización Automática

Para lograr la automatización del funcionamiento de una cámara es necesario trabajar con las imágenes proporcionadas por la misma. Existen diversas librerías de procesamiento de imagen por computador, ofreciendo solución a los problemas que serán descritos con posterioridad. Algunas de estas librerías son:

- «ImageMagic», enfocada al tratamiento de imágenes a través de un terminal.
- «Matio», permite ejecutar ficheros de Matlab (IDE ampliamente utilizado en el ámbito de la computación para el procesamiento de imágenes) de forma embebida en C++.
- «OpenCV», ampliamente disponible en un gran número de sistemas operativos y lenguajes de programación.

Para este proyecto se ha optado por usar la librería multiplataforma de acceso libre OpenCV, entre las arriba mencionadas, debido a que trabaja en ROS de forma nativa, reduciendo el coste de procesamiento. Así mismo, al estar disponible en diferentes lenguajes de programación, facilita enormemente el trabajo de experimentación con el sistema ROS, así como el desarrollo de proyectos mas aplicados en la industria utilizando sistemas como Windows, Windows Embedded, o Linux.

3.1.1. Control del Iris

La forma tradicional de calcular la luminosidad de una imagen es mediante su distribución de color (histograma). Calcular un histograma, almacenarlo, y compararlo con el histograma resultante de

la imagen modificada, supone un aumento considerable del tiempo de proceso.

En este trabajo se ha optado por desarrollar el cálculo del valor medio de los «pixels» de la matriz, cuyo resultado es comparado con un rango previamente establecido, para decidir si es necesario aclarar u oscurecer la imagen. Esto nos permite reducir el coste del proceso y, como se muestra en las figuras 4 y 5 (donde se comparan los resultados de ambos métodos), proporcionando una luminosidad de la escena óptima.

Figura 4: Aclarado de imagen. De izq. a dcha.: imagen original, cálculo de la media, ecualización de histograma

Figura 5: Oscurecido de imagen. De izq. a dcha.: imagen original, cálculo de la media, ecualización de histograma

3.1.2. Automatización del Enfoque

Cuando se habla del enfoque de una imagen, hay que tener en cuenta que por mucho que la imagen se encuentre perfectamente enfocada, éste no tiene por que ser siempre óptimo, ya que el área de interés puede no ser suficientemente extensa con respecto a la totalidad de la escena a tratar. Para hacer frente a este inconveniente, se han desarrollado dos utilidades:

- Enfoque automático, donde se hace uso del algoritmo de Sobel[14] para el cálculo del gradiente de la imagen a través de su mapa de bordes, el cual nos devuelve el gradiente de cada «pixel» de la misma, por lo que calculando su media podemos determinar el nivel de nitidez, comparando con el resultado obtenido en cada punto de enfoque desde el más próximo hasta el infinito, siendo el valor máximo el que determina el enfoque óptimo para el conjunto total de la escena. En la figura 6 se pueden observar las fases y el resultado de este procesos
- Infinito, se ha añadido la opción de poder cambiar la distancia focal de valor próximo a infinito y viceversa, con el fin de percibir con un nivel de nitidez satisfactorio un amplio rango de dimensión, a la vez que desactiva el enfoque automático.

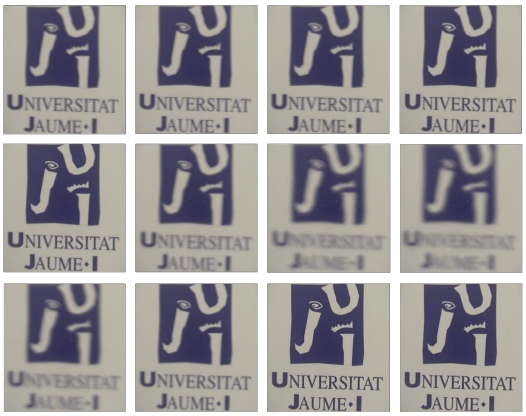

Figura 6: Transición en el proceso de enfoque automático

3.2. Detección de Objetos

Se testearon diferentes algoritmos de extracción de características con la intención de encontrar un algoritmo rápido y eficaz a la hora de reconocer objetos dentro de una escena, con el fin de desarrollar una herramienta que ofrezca la capacidad de indicar la posición y orientación de un objeto indicado mediante un patrón (imagen de muestra), y que a su vez pueda ser utilizado como base de un sistema de guiado de un ROV. Dentro de la extensa literatura relacionada con este tipo de algoritmos, podemos encontrar algunos realmente interesantes, como pueden ser:

- HARRIS[6], un detector de esquinas basado en los cambios de intensidad en distintas direcciones dentro de una misma región de la imagen. Presenta un nivel muy bueno en tiempos de ejecución, así como una gran precisión. A pesar de ello, no se ajusta a los requisitos necesarios, ya que nuestro dispositivo debe de poder ser capaz de trabajar bajo condiciones visuales extremas, el cual HARRIS no presenta tan buen comportamiento, aparte de no ser invariante a rotaciones y escalado.
- *Features from accelerated segment test (FAST)*[12], se basa en la vecindad de un punto candidato, donde se usa un umbral

fijado para comparar la intensidad de dicho punto con la de sus vecinos, y determinar si es o no un punto de interés. Es un algoritmo realmente rápido (como su nombre indica), a costa de la eficacia (su principal inconveniente). Es dependiente de la escala.

- *Binary Robust Independent Elementary Features (BRIEF)*[2], es un descriptor que necesita que se le proporcionen las detecciones (mediante FAST por ejemplo), haciendo uso de la distancia de Hamming para comparar los descriptores. Es un algoritmo muy rápido, pero no es invariante a las rotaciones, por lo que no satisface las necesidades del proyecto.

- *Scale Invariant Feature Transform (SIFT)*[7], se basa en la localización multiescala mediante una diferencia de gaussianas (DoG). Es invariante a las rotaciones y al escalado de los objetos de interés, representando la vecindad del punto de interés como un vector de características. Es el más preciso de todos los algoritmos utilizados, proporcionando el mayor número de características, pero penalizando en su tiempo de ejecución.

- *Oriented FAST and Rotated BRIEF (ORB)*[13], es un algoritmo que mezcla las características de los anteriores:

  - Se provee de puntos de interés haciendo uso de FAST.
  - Desconoce si esos puntos son buenos, por lo que hace uso de HARRIS para obtener la medida de esquinidad de cada uno de ellos, manteniendo como buenos los N de mayor valor.
  - Para resolver los problemas de dependencia de la escala, calcula una pirámide de imágenes (al igual que SIFT), obteniendo los puntos FAST de cada una de ellas.
  - Haciendo uso de los pares de puntos usados en BRIEF, ORB rota el patrón dependiendo del ángulo de la característica con el fin de solventar los problemas de invarianza a las rotaciones.

  Sus tiempos de ejecución son mucho mejores que SIFT, todo lo contrario que la robustez probada en nuestro sistema.

- *Speeded-Up Robust Features (SURF)*[1], basado en SIFT. Utiliza una aproximación básica de la matriz Hessiana (matriz cuadrada de n x n de las segundas derivadas parciales) para reducir el tiempo de computación. El determinante de la matriz Hessiana se utiliza para la localización de los puntos y para determinar la escala del objeto dentro de la escena.

  Para obtener la orientación de los puntos, se calcula el "Haar-wavelet" para las direcciones X e Y en una región circular de radio 6s, donde s es la escala (dimensión) del punto de interés. Wavelet [8] resuelve los problemas que presenta la Transformada de Fourier en el caso en que los puntos de interés se encuentren muy cerca unos de otros. A diferencia de Fourier, Wavelet utiliza un tamaño de ventana adaptado a las frecuencias. Con respecto a las imágenes, la Transformada de Fourier F(k) de una función f(x) de soporte finito se extiende entre [-infinito, +infinito]. Después de aplicar cualquier algoritmo de análisis a F(k), se pierde información al realizar la inversa de la transformada. En cambio, en la Transformada de Wavelet, la función y su transformada se encuentran en un intervalo finito, por lo que no hay pérdida de información al realizar la inversa de la transformada.

  Una vez calculados todos los vecinos, se estima la orientación dominante. Finalmente se suavizan los resultados mediante una Gaussiana.

  La característica principal de los puntos de interés SURF es la repetitividad. Si el punto es considerado fiable, el detector encontrará el mismo punto bajo distintos puntos de vista (diferente escala, orientación, etc.).

  Si SURF encuentra un número suficientemente grande de puntos que asegure la identificación del objeto buscado, lo dará por encontrado, aun estando parcialmente oculto, como se puede observar en la figura 7.

Después del estado del arte realizado para la elección del mejor algoritmo posible, donde a parte de los test realizados, se consideró el estudio de A.M. Romero y M. Cazorla [11] sobre las diferencias entre los algoritmos SIFT y SURF, comparando el número de puntos detectados y el tiempo invertido, obteniendo que SIFT es 3.39 veces más lento, encontrando 2.68 veces más puntos que SURF, se decidió implementar éste último.

3.2.1. Localización del Objeto

Se comenzó con la integración del algoritmo SURF dentro de un paquete de ROS (escrito en C++), donde SURF recibe un patrón y se suscribe a las imágenes publicadas por la cámara que esté siendo usada en ese momento (ver figura 8).

Este paquete publica un objeto de tipo mensaje «MultiArray» de flotantes, en el que almacenan las coordenadas del marco homográfico que define la posición y orientación del objeto. Haciendo uso de las coordenadas de los cuatro puntos que

Figura 7: Objeto encontrado parcialmente oculto

delimitan dicho marco, se calcula la intersección formada por las dos rectas generadas de la unión de los puntos opuestos del cuadrilátero. Esto nos indica la posición del objeto con respecto a la escena, y nos permite corregir su orientación con respecto a nuestro punto de vista.

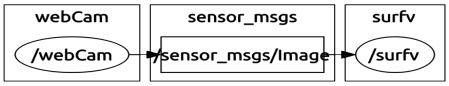

Figura 8: Diagrama ejecución de SURF en ROS

3.2.2. Surf en la Cámara Pan-Tilt

Después de la integración de la cámara y del algoritmo de SURF en el entorno ROS[4], e implementar una interfaz de alto nivel que permite su uso autónomo y/o supervisado, se ha tratado de integrar todas estas nuevas herramientas, con la intención de que sea la cámara, a partir de un patrón dado, capaz de ejecutar los comportamientos necesarios (rotación, enfoque, ecualización, etc.) para localizar el objeto deseado.

En esta ocasión, se ha hecho uso de una utilidad de ROS llamada «actionlib», la cual está constituida por:

- Servidor, ejecuta una petición solicitada por parte del cliente.
- Cliente, solicita al servidor que ejecute una acción.
- Fichero de especificación, almacena la definición de los objetos que se utilizarán: objetivos, resultados y realimentación de la acción.

Esta nueva herramienta desarrollada necesita que SURF esté en ejecución para poder funcionar, ya que responde a los valores publicados por el paquete "objectLocalization". Es por esta razón que se ha decidido la utilización de un «actionlib» en lugar de un servicio de ROS, ya que éste último no permite la suscripción a topics.

Una vez que el cliente lance la petición de ejecución de la acción, el «actionlib» comenzará a rotar la cámara en X e Y, tratando de cubrir la totalidad del rango de campo de visión de la misma. Mientras el objeto deseado no sea detectado, el «actionlib» mantendrá la cámara en movimiento. Una vez éste sea detectado, el «actionlib» dará por terminada su ejecución manteniéndose a la espera de una nueva solicitud.

4. Resultados

Se ha implementado una nueva interfaz de alto nivel, logrando un comportamiento autónomo de una cámara pan-tilt submarina, capaz de modificar sus parámetros logrando capturar imágenes con la mayor calidad posible, necesario para la búsqueda automática de objetos en el ámbito marítimo, donde el entorno se convierte en el mayor desafío.

Poniendo la mirada en los planes de contingencia, se ha optado por la inclusión del mando de la XBox, el cual permitirá tele-operar la cámara en el caso en que pueda aparecer un comportamiento incorrecto por parte de la misma. Del mismo modo, facilitará y dotará de mayor comodidad en las posibles tareas de tele-operación que puedan surgir por necesidad de la intervención a llevar a cabo. Dicha modalidad ya se encontraba integrada en ROS mediante el paquete **Joy**.

El esquema resultante de la integración de las nuevas utilidades se puede observar en la figura 9, donde los bloques destacados son el aporte de este trabajo.

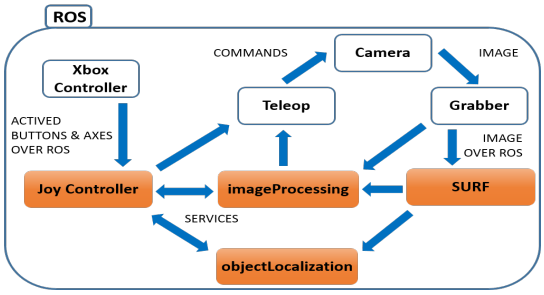

Figura 9: Esquema final del sistema

5. Conclusiones

Con el nuevo sistema desarrollado, se ha logrado la automatización de algunas funcionalidades importantes a la hora de montar la cámara en un ROV, dotando de independencia en el tratamiento de las imágenes capturadas.

Debido a la ausencia de retroalimentación por parte de la cámara, se realizó una batería de pruebas

con el fin de extraer los umbrales necesarios para cada una de las herramientas implementadas:

- Grado de luminosidad, se calculó el grado de luminosidad de imágenes tomadas en un entorno cerrado con luz artificial, comparando los resultados de luminosidad óptima extraída mediante histograma.
- Tiempo de enfoque, para el enfoque automático, se hace uso del tiempo que emplea la cámara en pasar del enfoque próximo al infinito (7.5 segundos, dato obtenido mediante mediciones en laboratorio).
- Patrones para SURF, deben de ser lo más pequeños posibles, sin exceder el límite que impida a SURF encontrar puntos de interés. Ese límite es dependiente de la resolución de la cámara, y se encuentra en torno a los 70x70 píxeles. Esto ofrece a SURF la posibilidad de identificar el objeto deseado a mayor distancia, ya que las dimensiones del patrón es el tamaño mínimo que debe de tener el objeto en la escena, siendo la escena el tamaño máximo del mismo.

La reflexión de la luz sobre los objetos dificulta la tarea de detección, ya que impide al descriptor extraer características del mismo.

Agradecimientos

Los autores agradecen el soporte del Gobierno Español, a través de los proyectos DPI2014-57746-C3 (MERBOTS) y DPI2017-86372-C3-1-R (TWINBOT), la Universidad Jaume I de Castellón con el proyecto P1-1B2015-68 (MASUMIA), y la Generalitat Valenciana (PROMETEO/2016/066).

Referencias

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[3] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[4] D. Fornas, J. Sales, A. Peñalver, J. Pérez, J.J. Fernández, R. Marín, and P.J. Sanz. Fitting primitive shapes in point clouds: a practical approach to improve autonomous underwater grasp specification of unknown objects. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(1-2):369–384, 3 2016.

[5] J.A. Goldbogen, B.L. Southall, S.L. DeRuiter, J. Calambokidis, A.S. Friedlaender, E.L. Hazen, E.A Falcone, G.S. Schorr, A. Douglas, D.J. Moretti, et al. Blue whales respond to simulated mid-frequency military sonar. *Proc. R. Soc. B*, 280(1765):20130657, 2013.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[7] Tony Lindeberg. Scale invariant feature transform. *Scholarpedia*, 7(5):10491, 2012.

[8] S. Mallat. Wavelets for a vision. *Proceedings of the IEEE*, 84(4):604–614, 1996.

[9] A. Peñalver, J. Pérez, J.J. Fernández, J. Sales, P.J. Sanz, J.C. García, D. Fornas, and R. Marín. Visually-guided manipulation techniques for robotic autonomous underwater panel interventions. *Annual Reviews in Control*, 40:201–211, 2015.

[10] J. Perez, J. Sales, A. Penalver, D. Fornas, J. J. Fernandez, J. C. Garcia, P.J. Sanz, R. Marin, and M. Prats. Exploring 3-D Reconstruction Techniques: A Benchmarking Tool for Underwater Robotics. *IEEE Robotics & Automation Magazine*, 22(3):85–95, 9 2015.

[11] A.M. Romero Cortijo, M. Cazorla, et al. Comparativa de detectores de características visuales y su aplicación al slam. 2009.

[12] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[14] I. Sobel, R. Duda, P. Hart, and J. Wiley. Sobel-feldman operator.